ASSO - Design the Internet

Homework 1

Team 33
Guilherme de Matos Ferreira de Almeida
João Pedro Carvalho Moreira
Jorge Daniel de Almeida Sousa
Lia da Silva Linhares Vieira
Nuno Afonso Anjos Pereira

Master in Informatics and Computing Engineering

12/02/2024

Contents

1 ABSTRACT

In this paper, we undertake the task of designing the Internet from scratch, envisioning a network architecture that anticipates the global scale and reliability demanded by millions of users. Drawing inspiration from the initial days of computer networking, we imagine a framework capable of accommodating diverse types of computers and seamlessly adapting to future technological advancements. Striving to minimize bias from the technological advancements we have today, we engage in a creative exercise aimed at fostering critical thinking and innovation in network design.

2 INTRODUCTION

With improvements in computing/processing power and breakthroughs in knowledge discovery, the dissemination of this knowledge to as many experienced minds as possible becomes imperative, with the hopes of even greater innovation and research. However, having a network that enables this sharing process requires more than networks restricted to the academic environment and isolated between institutions.

The goal is to have a network between networks: one might call it the **Inter-network network**, or **Internet** for short. This infrastructure should offer extreme reliability, ensuring that any major flaws are as transparent as possible to users connected. It should enable communication between devices across great distances (intercontinental at most) without requiring specialized knowledge beyond the devices' identifiers. It should also allow different types of computers with different hardware to connect to it.

3 SYSTEM REQUIREMENTS

Since we are aiming for global adoption, various critical requirements have to be met, some of which are:

- **Availability**: The system should ensure uninterrupted availability, minimizing downtime and service disruptions to provide consistent access to users worldwide.
- **Universal Connectivity**: The Internet must facilitate seamless communication and data exchange between diverse devices, regardless of their geographical location or underlying technology.
- **Scalability**: The architecture should be capable of accommodating exponential growth in users and data traffic without compromising the system's performance or stability.
- **Interoperability**: The Internet should support interoperability between different types of devices and systems, enabling them to communicate and collaborate efficiently.
- **Adaptability**: The system architecture should be flexible and adaptable to accommodate advancements in technology and evolving user requirements over time without requiring significant architectural changes.
- **Performance**: The Internet must deliver high-speed data transmission and low latency to ensure responsive and efficient user experience across various applications and services.
- **Standardization**: Standard protocols and interfaces should be adopted to promote compatibility, consistency, and ease of integration across different network components and devices.
- **Fault Tolerance**: Redundancy mechanisms and failover capabilities should be implemented to mitigate the impact of hardware failures, network outages and other potential disruptions on the Internet's operation.

Authors' address: Guilherme Almeida, up202006137@up.pt; João Moreira, up202003535up.pt; Jorge Sousa, up202006140@up.pt; Lia Vieira, up202005042@up.pt; Nuno Pereira, up202007865@up.pt.

4 SYSTEM OVERVIEW AND ARCHITECTURE

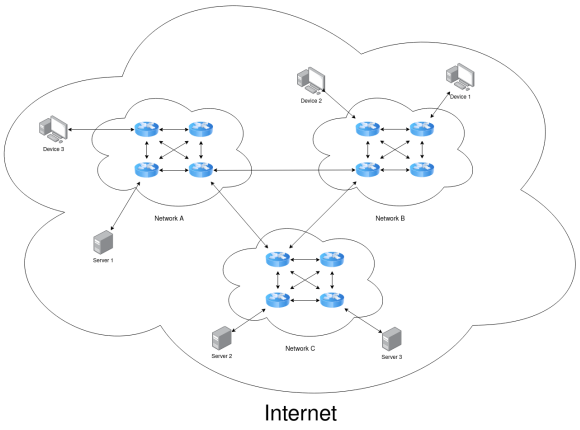

Fig. 1. High-Level Internet Overview

We conceptualized the internet as a hierarchical network, composed of smaller networks nested within larger ones as depicted in Figure 1. This hierarchical structure facilitates efficient routing of data across the globe since smaller networks don't need to keep information about other smaller networks physically distant from them. This would be the job of the higher levels of the network. This means that each level only needs to store the information necessary to route internet traffic to its network subsection (including "children" sub-networks and connected devices). Further optimizations can be implemented so that each level only needs to keep track of its direct descendants, further reducing the amount of information stored in each (sub-)network. Due to the above, and because each network subsection has sufficient information to route all internet traffic within itself, communications that originate and end in the same sub-network never have to reach higher levels of the network hierarchy. This reduces the time required for a message to depart from its origin and reach its destination, as well as not overloading that network's "parent network" with unnecessary traffic.

As mentioned in Section 3, various challenges need to be addressed before we can call this "The Internet".

4.1 Reliability and Availability

Having multiple networks connected allows, for instance, in the scenario presented in Figure 1, Device 3 connected to Network A to access Server 3, located within Network C. However the internet

must be reliable to guarantee communication even in the presence of failures. Our approach for achieving this reliability involves implementing redundancy, meaning that the connection between two networks must be redundant; there must be more than one path from one network to another to allow redirecting network traffic through a different route in the event of network partitions.

For simplicity purposes, this redundancy is not illustrated in Fig. 1 and is explained in Section 4.6.

4.2 Addressing

In order for computers to communicate, there must be a way for them to address each other, just as people know the address of the recipient when sending a letter. These addresses should uniquely identify each connected device.

When facing this challenge, we considered two potential approaches: a more naive one involving Geographical Identifiers with some inherited problems and a more standard one utilizing a Universal Identifier.

4.2.1 ***Geographical Identifier***

Dividing the world into continents, then further dividing these into countries, and breaking down countries into smaller regional divisions, each of these divisions forms a part of the hierarchical structure of the Internet.

Every computer connected to the Internet is assigned an Unique Identifier. This identifier is crucial for routing data across the network.

The structure of this network device identifier is as follows:

<continent>-<country>-<region>-<serial id>

where the **<serial id>** is an increasing identifier.

For example, let's consider a computer located in the northern region of Portugal. Its unique identifier might be structured as follows: EU-PT-NRT-1.

How to send a message from identifier A to identifier B?

```
$$EU-PT-NRT-1 (PC1) \rightarrow AM-EUA-FL-1 (PC2)$$
```

PC1 sends a packet to the router it's connected to (router of the northern region: EU-PT-NRT). This message contains the destination identifier of PC2. By inspecting the destination identifier, the receiving router is capable of redirecting the message to a new router. For this example, the router of the North region wouldn't be able to redirect directly to a router/PC satisfying AM-EUA, thus it would have to ascend in the network level and send it to the EU-PT router. This router is capable of sending it to an AM-EUA router through submarine cables. From this level, this router is already in the destination country, thus it needs to descend in the network hierarchy to send it to the region router and from there to PC2.

4.2.2 Problems with this approach

While this solution may seem simple enough to implement at first glance, there are some crucial flaws that prevent it from being a viable solution:

- Different geographical regions are hierarchically differently structured: the *<continent>-<country>-<region>-<serial id>* model closely reflects the organization of the network infrastructure itself, with sections nested within sections. However, not all regions of the globe share the same geopolitical granularity. One example is the comparison between Portugal (which consists of regions and cities) and the USA (comprising states, counties, and ultimately cities). Modeling a standard representation of Universal Identifiers that could represent each and every device across the globe would become a cumbersome task.
- Non-semantic way to represent different levels of the network: while the aforementioned model may work for identifying specific devices, it fails to represent in a semantic way the higher-level structures present. It does not make sense for a country-wide router to have a regional identifier. A fix for this could be to simply drop that component from the address altogether but then we would lose the benefits of every device and piece of infrastructure having a standard way of being addressed.
- Varying size/Unnecessary size for identifying addresses: like any computer system, these addresses need to be presentable at a physical level, that is, they need to be able to be serialized into bytes that can be processed by a computer. Following the ASCII standard, the - character is a fixed length. The *continent* identifier can be limited to 3 *bits* if it is represented as a number (there are at most 7 continents, depending on who you ask) or 2/3 ASCII characters (following the examples given), which are 8 *bits* in length each. The amount of space needed to encode the *country* should be carefully decided considering that, in 1970, there were 191 recognized sovereign states in the world. Assuming the most space-efficient representation is used, this corresponds to an *8-bit* number. If we decide to use text, this number goes up to 16/24, depending on the number of characters present. This thought process can also be applied to the region component of addresses. Finally, there's the *serial* component, which is an arbitrarily large number. On some networks there might be a small number of connected devices so the size of this component can be limited to a few *bits*. On other networks there might be thousands or millions of devices connected, in which case more space is needed. This uncertainty imposes that a sufficiently large size should be chosen to accommodate every possibility which, as we just saw, might be unnecessary.

All these points make us believe that a better, more well-thought representation should be designed in order to address this issue. We can take inspiration from the previous representation, namely in the *serial* component which, being a number, can encode a larger amount of addresses than typical textual representations.

4.2.3 ***Universal Identifier***

With the assumption that each individual will have a single device with connectivity capabilities, the ideal outcome is for each device to have a unique address on the network. By the end of the 1960's the world's population neared around 4 billion people. Since numbers are encoded in bits, "prettier" representations are the ones that are a power of 2. The next power of 2 after 4000000000 is 4294967296,

corresponding to 2^{32}. The previous assumption only takes into account personal devices but when we look at the numbers we realize that there would be close to 300 million addresses left for non-personal machines, which is easily manageable.

This way, we can design an address representation utilizing 32-bit numbers. One drawback of this method is the tedium of writing 32 digits, but this can be easily mitigated by recognizing that a 32-bit number spans 4 bytes. This way, instead of writing 32 binary digits, we can write 4 different bytes (0-255) and achieve the same value. For simplicity, we can add a separator between each pair of bytes, for example a . character:

XXX.XXX.XXX.XXX

We can immediately identify some advantages of this format compared to the previous one:

- There is no distinction between a user device and a "network device" (like an enterprise router), allowing the network infrastructure to be built on top of existing communication interfaces (canonically called a "link layer").
- The different organizational levels that different countries might have are discarded: one simply needs to have a network address to be reachable, independently of where they are located.

Some problems might appear as the world's population grows but there are techniques that can be employed to virtually simulate more distinct addresses than there really are. One of those techniques is the possibility for a router to mask all addresses on "one side" under a single address on the "other side", following a Many-To-One semantic. Details on how to implement this are not under the scope of this document.

4.3 Interoperability between different systems and devices

The purpose of the Internet is to serve as a network of networks: conceptually its purpose is to connect multiple networks together.

The problem of establishing a network between several devices has already been solved. This way, it is not necessary for us to specify anything with regard to this. By using the infrastructures and technologies already provided by local networks and link-local technologies and protocols we can compose several local networks to achieve our original goal.

However, the need for network-related hardware and software to speak a unified "language" still holds and allowing for multiple link-local protocols introduces too much entropy in the functionalities needed to perform network-level operations. As such, we can design a new information unit related to the network side of the infrastructure, which we will call *packet* for convenience and which is described in sub-section 4.4. By using this structure as the payload for link-local protocols we successfully abstract the underlying connection from the network layer of the infrastructure, which just needs to deal with these packets after they are extracted from the link-local structures, ensuring that network operations perform independently of the device.

4.4 Packets

As mentioned in the sub-section 4.3, one can encode information related to network operations inside lower-level constructs provided by already existing communication infrastructures (for example, this network information can be the data payload of an Ethernet frame) using *packets*.

When one network-capable computer attempts to communicate with another, the former must know the latter's Network Address, defined in section 4.2, hinting at this field having to be present in each packet. For convenience, the senders address should also be present. Also, the payload data can have a varying size. There are 2 *naïve* ways to tackle this issue:

- Enforce a fixed size for all packets, providing an upper bound for the amount of information that can be sent in each packet.
- Have a new field in the packet structure that specifies the length of the data being transmitted.

These aren't without flaws, of course: the first approach can lead to lots of empty space in a packet that can slow down transmission, since each packet should be a whole unit of information with respect to the network; as for the second approach, the size being completely dynamic means that there is no limit to how much information can be transmitted, which also has the disadvantage of not knowing how many *bits* of information are needed to encode the length of the packet payload.

A better approach takes inspiration from both methods: have a packet field specifying the payload length but restrict this length to an upper bound. This upper bound should be defined according to extensive experimentation to figure out the optimal size needed, but since we are proposing something new we will define a set value: $2^{16} - 1 = 65535$. Using this value, we can make the size of the length component be fixed at 16 *bits* (2 *bytes*) long and the payload can be anywhere between 0 and 65535 *bytes* in length.

The addresses and payload length compose what is called the *header*.

Adding another 16 *bits* of padding allows not only for the header information to be 32 *bit* aligned but also gives room for further extensions and revisions of the protocol, like flags or extra fields.

This means that the structure of a network packet is, at least, as follows:

```
|  Sender Network Address  |
| Receiver Network address |
| Payload Length | Padding |
|          Data ...        |
```

4.5 Routing

For the computers to communicate with each other, there must be some type of connection between them. The simplest case would be a point-to-point connection between two devices, such as in Figure 2.

Fig. 2. Point-to-Point Connection

By using the same principle in a network made of various devices, we would end up with a connection for each device pair (Figure 3)

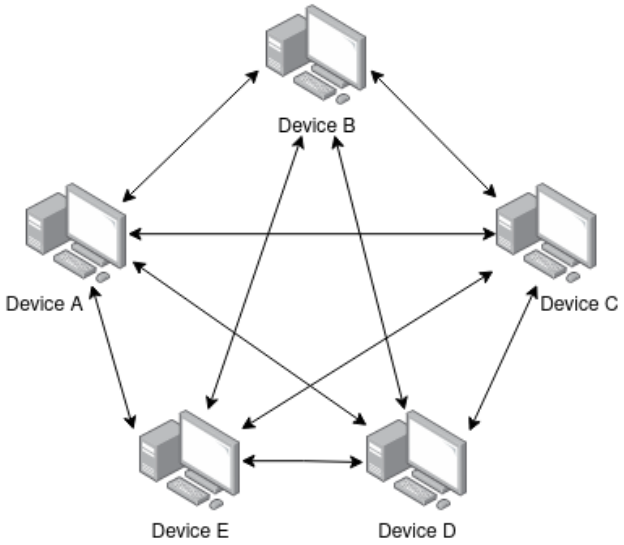

Fig. 3. Ring Connection between devices

To solve that we firstly need to introduce the concept of a router.

4.6 Routers

A router is a physical device that plays a crucial role in the functioning of the Internet. The primary function of a router is to determine the best path for forwarding data packets to their destination. This process, known as routing, involves analyzing the destination Universal Identifier of incoming packets and consulting a routing table to determine the next hop along the path. The routing tables are explained in the sub-section 4.7.

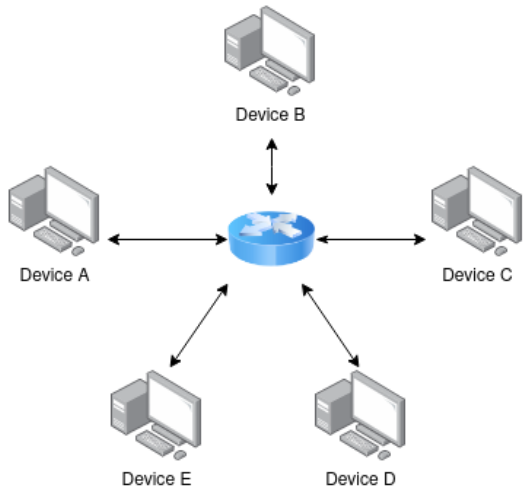

Fig. 4. Connection between devices with routers

In the simplified Internet model present in Figure 1 we can notice that there are routers connecting the various networks. This brings some possible questions, such as:

- What would happen in case of failure of one of these routers?
- What if there is a lot of congestion in the connection between two networks?

To ensure reliable connectivity between networks and some fault tolerance for failing routers, we must ensure that there is no single point of failure in our model.

When confronted with this problem, we contemplated two potential approaches:

4.6.1 *Leader-Follower Algorithm*

Our first idea is to have a leader-follower algorithm, where we can think of a small network constituted of routers as a cluster of leader and follower nodes. The leader would periodically broadcast a small message to the follower nodes and, if the majority of the nodes received this message the cluster could assume that the leader is functioning correctly.

If the cluster concluded that there was a problem with the leader node, the remaining nodes would start an election for a new leader using some variant of a consensus algorithm [1]. This would ensure that there was always a valid node to be the leader and communicate with other networks, as long as the majority of the nodes were functional.

4.6.2 *Problems with this approach*

This approach brings various problems:

- This does not solve the scenario where the connection is congested.
- If the current leader experiences a system failure (effectively leaving the router cluster with no leader), the followers have to initiate a leader-election process. Depending on the algorithm employed this process can be more or less slow, causing the whole network served by the router cluster to lose connectivity to the rest of the Internet while the election process takes place.
- For this solution to be viable the majority of the nodes need to be in a healthy state, which is not always the case.
- Most consensus algorithms bring a lot of time and spatial complexity and this can bring a huge overhead to this routing mechanism in which both those criteria are critical and sensitive.

4.6.3 *Redundancy*

To deal with the previous problems we tried to keep things simpler and came out with an approach where we would simply establish redundant connections between routers, either directly or through intermediary nodes (routers). This redundancy ensures that if one path becomes unavailable due to a hardware failure we can find an alternative path, painting the continuous connectivity between networks. In contrary to the other approach, this can also deal with scenarios of network congestion, since we can easily apply a multi-criteria algorithm to find the best routes according to various parameters (number of hops, connection delay, cost/weight, etc.).

This approach is crucial for maintaining network reliability and fault tolerance and it can easily solve a scenario of large-scale networks where the failure of a single router could disrupt the connectivity of many devices. By having redundant paths, routers can effectively manage traffic flow and minimize downtime, ensuring seamless communication between networks.

4.7 Routing Tables

A routing table is a structure present in any network capable device that contains enough information for a received packet to be correctly forwarded through a network until it reaches its desired destination.

4.7.1 ***Full Routing Information***

A naïve approach would be to store an entry for every Unique Identifier which is of interest and the associated network interface through which traffic should flow.

In this case, the structure of the routing table could be as follows:

```
| Destination | Interface |
| 192.168.0.0 |    if1    |
| 10.32.26.1  |    if2    |
```

4.7.2 ***Problems with this Approach***

This method has one obvious flaw and that is the amount of space needed to store the information for routing traffic to other parts of the network.

If many of the entries in the routing table are related to contiguous addresses, one optimization that can be made is to group these addresses into a range and only store a representation of this range, drastically reducing the number of entries needed to store the full routing information.

4.7.3 ***Address Masking***

Since an address range is a sequential set of addresses, and since addresses are basically 32 *bit* integers, many of these addresses share *bit* sequences in common. For example, 255.255.255.255 and 255.255.255.254 only differ in the last *bit*. Another way to see this is that there is a certain amount of *bits* that stays the same between both addresses, in this case, the first 31 *bits*.

If we were to place these 31 bits "on top" of a sequence of 32 0 *bits*, we would get what is called the *base address* for a given address range, in this case 255.255.255.254.

Since we admit both addresses are in the given address range, we also admit that both are representable by the given *base address*. In order for both addresses, in this case, to derive the *base address*, we can apply a bit-wise AND operation with a number that is originated by 31 1 *bits* and 1 0 *bit*, from highest to lowest, which would look like this: 11111111111111111111111111111110. In general, for any set of addresses, once we find the number of common *bits* in them, if we apply the same algorithm to generate a string of bits which are bit-wise AND'ed with any given address we will obtain the *base address* for that address range.

This special string of *bits* is called a *network mask*, or *netmask* for short. And since it is also 32 *bits* long, it can also be represented as an address.

This way, we can compact a large number of addresses into just 2: the *base address* and the *netmask*.

After these optimizations an example routing table could look like this:

```
|  Destination  | Netmask       | Interface |
| 192.168.0.0   | 255.255.255.0 | if1       |
```

Although we store more information per entry, each entry has the ability to represent more addresses, effectively reducing the amount of space needed to store all the routing information needed for a device.

Considering the routing table above, a packet with a destination address of 192.168.0.1 would match the entry in the table, because if we calculated the bit-wise AND operation between the packet's destination address and the entry's *netmask*, the result would be equal to the entry's destination address in the table and the packet is forwarded through interface if1.

If a packet had a destination address of 192.167.0.1, the *bit-wise* AND operation between this address and the network mask would result in a value different from the one in the table, meaning that the packet would not be routed through the address specified at that entry.

In the case of multiple matches, the entry with the highest value for the *netmask* is chosen: since the address range it represents is smaller, this entry is more specific.

4.7.4 ***Further optimizations***

One more optimization that could be made to decrease the space needed per entry is to store the amount of *bits* set to 1 in the *netmask* instead of the entire *netmask*, since one can be calculated from the other and vice-versa.

This, however, would lead to having to actually perform that calculation, which is fast for a single entry but slow when scaled to a lot more. Improvements in computing power can mitigate this issue, making this optimization a good candidate for implementation in the long term.

5 PRACTICAL APPLICATIONS

Following the architecture described in this document, a number of practical applications can be implemented, ranging from providing internet access as a billable service to serving a global interconnected knowledge base.

5.1 Internet as a Business Model

Although networks can be infinitely composable, there is a certain level where we can encounter all kinds of "user networks" like domestic or corporate networks.

This has brought about the existence of companies that control the networks directly above these which sell access to their part of the infrastructure as a paid service.

Since these companies are providing access to the Internet they can be called Internet Providers or Internet Service Providers (ISP).

5.2 Internet as an Engine for Sharing Knowledge

Having all devices connected through the Internet (possibly) allows every device to share any content it intends to any other device.

With a bit of ingenuity, this can allow for the deployment of devices whose sole purpose is to serve as an open repository of knowledge. Having pieces of software built for navigating this "web" could allow for better information and knowledge sharing.

6 CONCLUSION

This paper lays out a blueprint for designing the Internet from scratch, envisioning a network architecture that anticipates the global scale and reliability demanded by millions of users. Our approach involved creating a hierarchical network structure, composed of smaller networks nested within larger ones, to efficiently route data across the globe. We addressed key system requirements such as universal connectivity, reliability, scalability, interoperability, adaptability and accessibility.

One of the critical aspects tackled was the addressing scheme for devices connected to the Internet. Two approaches were explored: a hierarchical addressing system based on geographical regions and a universal identifier system using 32-bit numbers. The advantages and limitations of each approach were identified, ultimately advocating for the adoption of a universal identifier system to overcome the challenges posed by geopolitical differences and scalability concerns.

Furthermore, we discussed the concept of routing and the role of routers in directing data packets across the network. We outlined the importance of routing tables in facilitating efficient data transmission and explored the structure of network packets.

The proposed architecture opens up possibilities for practical applications, such as internet connection services provided by ISP's, leveraging the flexibility and scalability of the designed network infrastructure.

REFERENCES

[1] Consensus (computer science). https://en.wikipedia.org/w/index.php?title=Consensus_(computer_science)&oldid=1189781469. [Online; accessed 11-February-2024].